三种求解微分方程数值解的方法

2022 年 3 月 19 日

目录

1 前言

对于绝大多数微分方程,我们无法求出解析解。下面介绍三种针对问题

$$y' = f(x, y) \tag{1.1}$$

的求数值解的方法。

2 欧拉法

2.1 算法公式

$$\begin{aligned} \omega_0 &= y_0 \\ \omega_{i+1} &= \omega_i + hf(t_i, \omega_i) \end{aligned} \tag{2.1}$$

2.2 代码示例

Listing 1: **main.m**

```
%%
% $$y_{k+1}=y_k+hf(x_k,y_k)$$

clc,clear,close all
n = 100;
f = @(x,y) -20*x;                % 设置微分方程
a = 0;                           % 区间左端点
b = 1;                           % 区间右端点
h = 1/n;                         % 取点间隔
y0 = 1;                          % 设置初值

y = euler(f,a,b,h,y0);
plot(a:h:b,y)
```

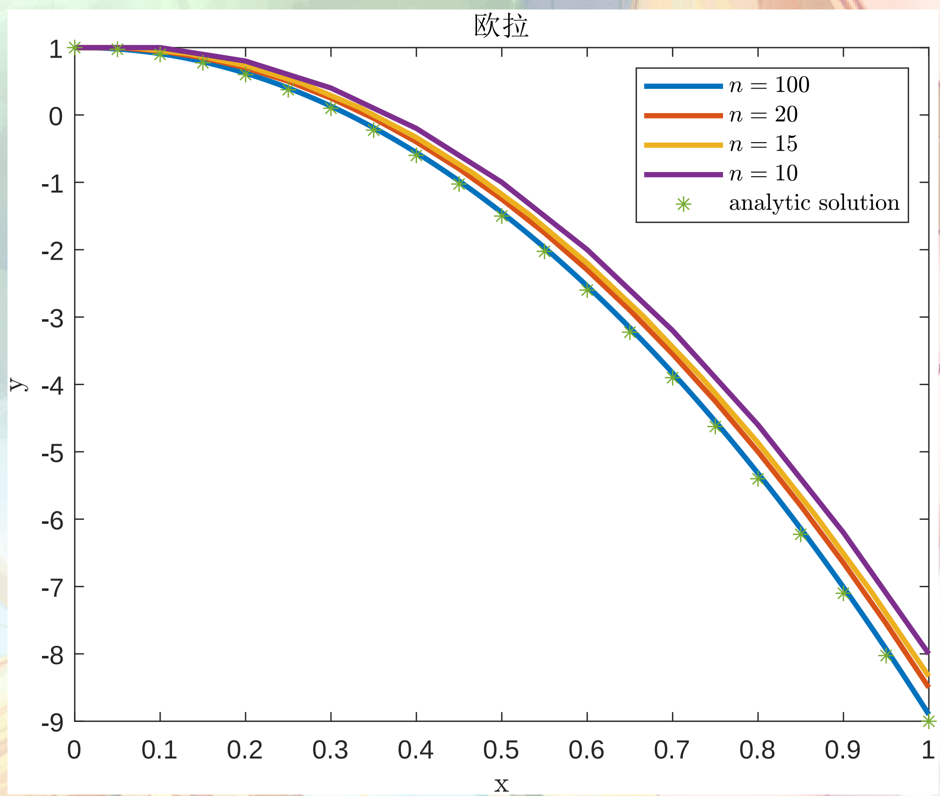
Listing 2: **euler.m**

```
function y = euler(f,a,b,h,y0)
x = a:h:b;                       % x_i
n = length(x)-1;                 % n
y = zeros(1,n+1);                % 预分配内存
y(1) = y0;                       % 设置初始值
for i = 1:length(x)-1
    delta = f(x(i),y(i));
```

```
    y(i+1) = y(i)+delta*h;
end

end
```

2.3 结果展示



可以看出，随着步长 h 的不断减小，数值解也越来越精确。

3 改进的欧拉法

3.1 算法公式

$$\omega_0 = y_0 \omega_{i+1} = \omega_i + \frac{h}{2}(f(t_i, \omega_i) + f(t_i + h, \omega_i + hf(t_i, \omega_i))) \tag{3.1}$$

3.2 代码示例

Listing 3: **main.m**

```
clc,clear,close all
```

```
n = 100;
syms x y
dif_f = @(x,y) -20*x;
h = 1/n;
a = 0;
b = 1;
y0 = 3;
x = a:h:b;

y = euler_improve(dif_f,y0,x);
plot(x,y,'LineWidth',2)
```

Listing 4: **euler_improve.m**

```
function y = euler_improve(dif_f,y0,x)
h = x(2)-x(1);
y = zeros(1,length(x));
y(1) = y0;
for i = 1:length(x)-1
    temp = y(i) + h*dif_f(x(i),y(i));
    delta = (dif_f(x(i),y(i))+dif_f(x(i+1),temp))/2;
    y(i+1) = y(i)+delta*h;
end
end
```

3.3 结果展示

相比较于欧拉法,改进的欧拉在步长 h 较大时就已经取得了非常好的精度(以至于四条曲线几乎重合)。

4 后退欧拉法

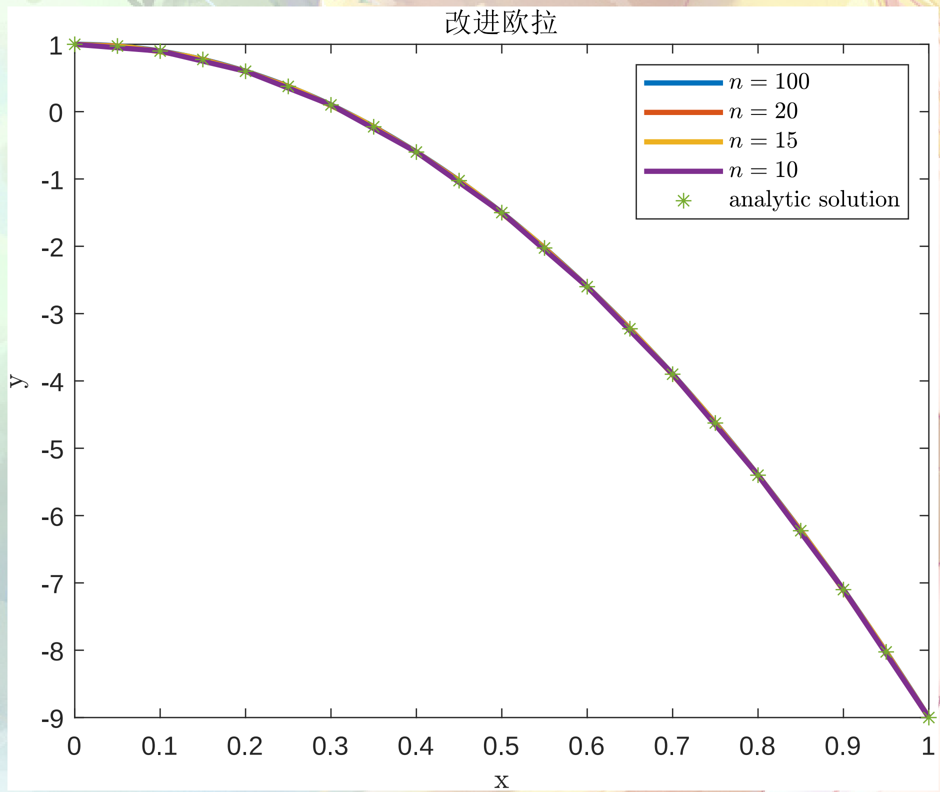
4.1 算法公式

$$\begin{aligned} \omega_0 &= y_0 \\ \omega_{i+1} &= \omega_i + hf(t_{i+1}, \omega_{i+1}) \end{aligned} \tag{4.1}$$

4.2 代码示例

Listing 5: **main.m**

```
clc,clear,close all
```




```
syms x y
dif_f = @(x,y) -20*x;
n = 100;
h = 1/n;
a = 0;
b = 1;
y0 = 1;
x = a:h:b;

y = euler_back(dif_f,y0,x);
plot(x,y,'LineWidth',2)
hold on
```

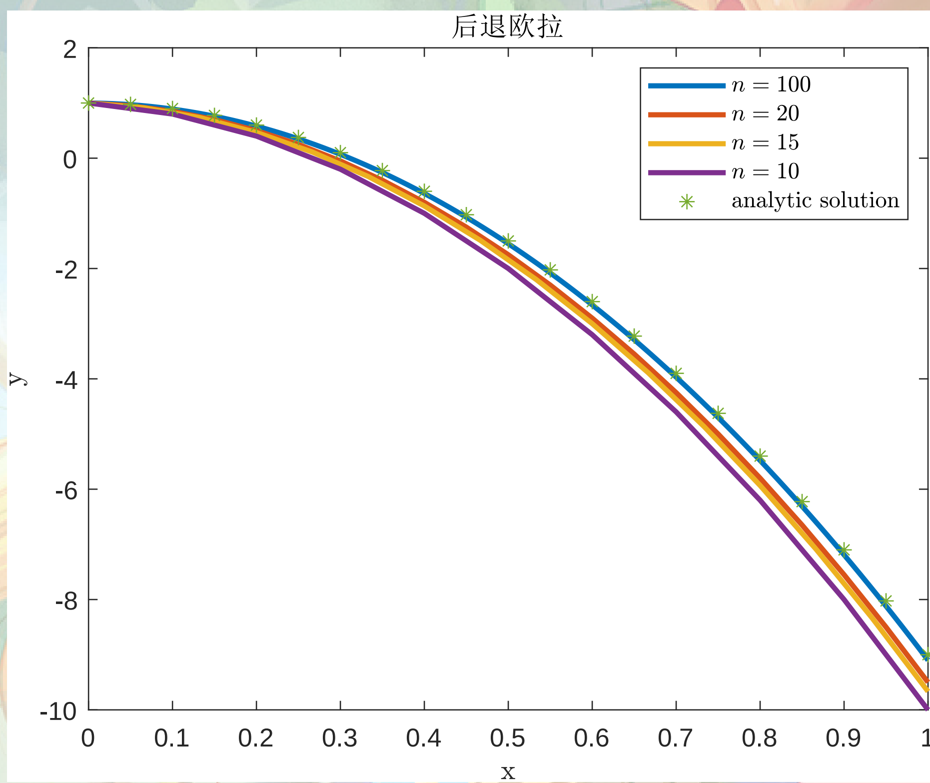
Listing 6: **euler_back.m**

```
function y = euler_back(dif_f,y0,x)
h = x(2)-x(1);
y = zeros(1,length(x));
y(1) = y0;

k = 10;
```

```
for i = 1:length(x)-1
    yy = y(i) + dif_f(x(i+1),y(i))*h;
    for j = 1:k
        yy = y(i) + dif_f(x(i+1),yy)*h;
    end
    y(i+1) = y(i) + dif_f(x(i+1),yy)*h;
end
end
```

4.3 结果展示



后退欧拉法是一种隐式的方法，在编程过程中，需要利用迭代法来求解一个一元方程。相比较于显示方法，隐式方法允许在相对较大的步长中具有足够的误差控制，以及更好的效率。